ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR TEKNIK

Lösningarna till tentamensuppgifterna sätts ut på kurssidan på nätet i dag kl 13.

Denna tenta kommer att vara färdigrättad Ti 26/10 och kan då hämtas på mitt tjänsterum T2221 mellan kl 13 och 15. Vid detta tillfälle har du också möjlighet att diskutera rättningen. Tentor som ej hämtats då placeras på studentexpedition.

Tentamen i Objektorienterad Programmering 5p, Au, D, E, Fri, Pr, 041020

Hjälpmedel : Inga
Tid : 08 - 13
Ansvarig lärare : Gunnar Joki Tel : arb 303317
hem 274825
mob 070-5474825

Svar till samtliga uppgifter 1-15 ska skrivas på utdelat extra papper. Använd ett papper till uppgifterna 1-5, två papper till uppgifterna 6-10 och ett papper per uppgift till uppgifterna 11-15 Skriv din tentamenskod på varje inlämnat extra papper.

Den maximala poängen för respektive uppgift står angiven efter uppgiftens nummer. Totalt kan 40 poäng erhållas. För betyget 3 krävs ca 20, för betyget 4 ca 28 och för betyget 5 ca 34 poäng.

Om inget speciellt anges gäller frågorna Borland C++.

Detta häfte ska du behålla.

Lycka till!

1) (1p) I specifikationen för klassen PERSON finns de privata medlemsvariablerna persnr och namn definierade enligt:

```
char persnr[12];  // Personnummer
char namn[30];    // Namn
```

Implementera konstruktorn nedan så att den kopierar parametervärdena till motsvarande medlemsvariabler.

```
PERSON(char *persnr = "", char *namn = "");
```

---

2) (1p) Skriv de satser som skapar ett personobjekt person av klassen PERSON ovan, med personnumret 111111-1111 och namnet Kalle Anka.

---

3) (1p) Flickor har en jämn näst sista siffra i personnumret emedan pojkar har en udda. Implementera medlemsfunktionen is_girl i klassen PERSON med nedanstående specifikation, så att den returnerar true om flicka annars false.

```
bool is_girl();
```

---

4) (1p) Man kan istället för en medlemsfunktion använda sig av en friendfunktion för funktionen is_girl enligt upgift 3 ovan. Skriv specifikationen för en sådan friendfunktion.

---

5) (1p) Istället för att ha namnet skriv på medlemsfunktionen, i uppgift 6 nedan, kan man använda sig av en överlagrad utskriftsoperator-funktion. Skriv specifikationen för denna.

---

6) (2p) Klassen PERSON avbildar personer enligt:

```
// Specifikation av klassen PERSON – Person.h

class PERSON
{
   private:
      char persnr[12];
      char namn[30];
   public:
      PERSON(char *persnr = "", char *namn = "");
      void las();       // Läser personens data med ledtexter
      void skriv();     // Skriver personens data med ledtexter
      int get_alder();  // Returnerar åldern för personobjektet
      bool is_girl();   // Returnerar true om flicka annars false
      bool same_sex(PERSON p); // Returnerar true om objekt och
                               // parameter har samma kön annars false
};
```

Implementera medlemsfunktionen las.

7) (2p) Implementera medlemsfunktionen skriv i klassen PERSON ovan så att den skriver ut personens data.

---

8)(2p) Skriv ett huvudprogram som skapar och läser in data till en person av klassen PERSON ovan och skriver ut om personen i fråga är en pojke eller en flicka.

---

9) (2p) För att kunna kontrollera om två personer har samma kön finns i klassen PERSON ovan medlemsfunktionen same_sex. Implementera denna medlemsfunktion så att den

returnerar true om objektet har samma kön som personen p, annars false.

---

10)(2p)Implementera medlemsfunktionen get_alder i klassen PERSON ovan om dagens datum, date fås som en sträng på formen ”20041023” med den givna funktionen:

```
void get_date(char *date);
```

Personer som är över 100 år har ett plustecken istället för ett minustecken i sitt personnummer.

---

11)(5p)En motorventil består av en motor och en ventil. Implementera medlemsfunktionerna för klasserna MOTOR, VENTIL och MOTORVENTIL enligt nedanstående specifikationer. Skriv sedan ett huvudprogram som skapar ett objekt av klassen MOTORVENTIL med en motor som gör 20 varv/min och en ventil som öppnar sig helt efter 50 varv och skriv ut den totala öppningstiden för motorventilen.

```
//Specifikation av klassen MOTOR - Motor.h

class MOTOR
{
   private:
      float varvtal;    // Motorns varv per minut
   public:
      MOTOR(float varvtal = 0);
      float get_varvtal();// Returnera varv per minut
 };


//Specifikation av klassen VENTIL - Ventil.h

class VENTIL
{
   private:
      float antal_varv; // Antal varv för att öppna ventilen helt
   public:
      VENTIL(float antal_varv = 0);
      float get_antal_varv(); // Returnera antal varv
};


//Specifikation av klassen MOTORVENTIL - Motorventil.h

#include "Ventil.h"
#include "Motor.h"
```

```
class MOTORVENTIL
{
   private:
      MOTOR m;
      VENTIL v;
   public:
      MOTORVENTIL(float varvtal = 0,float antal_varv = 0);
      float get_tid(); // Tiden i minuter för att öppna ventilen
};
```

---

12)(5p)Klassen PERSONAL avbildar anställda personer på ett företag enligt:

```
// Specifikation av klassen PERSONAL – Personal.h

#include ”Person.h”

class PERSONAL : public PERSON
{
   private:
      float lon;  // Månadslön
   public:
      PERSONAL(char *persnr = ””, char *namn = ””, float lon = 0.0);
      void las();       // Läser data med ledtexter
      void skriv();     // Skriver data med ledtexter
      float get_lon();  // Returnerar lön
};
```

Implementera konstruktorn och funktionerna las, skriv och get_lon och skriv ett huvudprogram som läser in två personal och skriver ut dem i ordning efter lön med den som har lägsta lönen först.

---

13)(5p)Ett företag har all sin personal enligt uppgift 12 ovan i textfilen Personal.txt med personnummer, namn och lön radvis enligt:

```
111111-1111 C.C 25000
121212-2222 A.A 18000
. . . . . .
```

I sin jämställdhetsplan vill företaget skriva in vad kvinnorna tjänar i medel och vad männen tjänar i medel. Skriv ett huvudprogram som läser hela filen Personal.txt och skriver ut kvinnornas och männens lönemedelvärden.

---

14)(5p)Biosalonger kan avbildas som objekt av klassen BIOSALONG enligt:

```
// Specifikation av klassen BIOSALONG – Biosalong.h

class BIOSALONG
{
   private :
      int antal;        // Antal platser
      bool *stolar;     // Pekare till första stolen i salongen
      char namn[20];    // Salongens namn
   public:
      BIOSALONG(int antal = 0, char *namn = ””);
```

```
        // Skapar dynamiskt en biosalong som heter namn och har antal
        // st platser i form av en vektor stolar, där alla platser är
        // lediga och har värdet false.

        ~BIOSALONG();
        // Avallokerar alla platser

        bool boka(int radnr, int platsnr);
        // Om platsnr på radnr är ledig markeras den som upptagen med
        // true och funktionen returnerar true annars false.

        void visa();
        // Visar salongens platser med 10 stolar i varje rad och där de

        // lediga platserna markeras med 0 och de upptagna med X.

      int get_antal();   // Returnerar antalet platser i salongen
};
```

Implementera klassens medlemsfunktioner så att huvudprogrammet nedan bokar lediga platser i salongen.

```
#include "Biosalong.h"
#include <iostream.h>

void main()
{
   BIOSALONG bio(40, "Amanda");
   int radnr, platsnr;

   bio.visa();
   cout << ”Vilken plats vill du boka? ”;
   cout << ”Ange rad (1 – ” << bio.get_antal()/10 << ” uppifrån ): ”;
   cin >> radnr;

   while (radnr != 0)
   {
      cout << ”Ange plats på denna rad (1 – 10 från vänster) : ”;
      cin  >>  platsnr;
      while (!bio.boka(radnr, platsnr))
      {
         cout << "Platsen är upptagen! " << endl;
         cout<<”Ange rad (1 – ”<<bio.get_antal()/10<<” uppifrån ): ”;
         cin >> radnr;
         cout << ”Ange plats på denna rad (1 – 10 från vänster) : ”;
         cin  >>  platsnr;
      }
      bio.visa();
      cout << ”Du har bokat plats ” << platsnr << ” rad " << radnr;
      cout << ”Vilken plats vill du boka? ”;
      cout <<”Ange rad (1 – ”<<bio.get_antal()/10<<” uppifrån ) : ”;
      cin >> radnr;
   }
}
```

---

15)(5p)Klassen BIOSALONG ovan måste ha egen kopieringskonstruktor och tilldelningsoperator för att åstadkomma verkliga kopior och tilldelningar för objekten. Skriv specifikationer för kopieringskonstruktorn resp. tilldelningsoperatorn och implementera dessa.

Lösningar till tentamen i Objektorienterad Programmering 5p, 041020

1)
```
PERSON::PERSON(char *persnr, char *namn)
{
   strcpy(this->persnr,  persnr);
   strcpy(this->namn,  namn);
}
```

2)
```
PERSON person(”111111-1111”, ”Kalle Anka”);
```

3)
```
bool PERSON::is_girl()
{
   return (persnr[9] % 2) == 0;
}
```

4)
```
friend bool is_girl(PERSON p);
```

5)
```
friend ostream &operator<<(ostream &ut, PERSON p);
```

6)
```
void PERSON::las()
{
   cout << "Ge personnummer : ";
   cin >> persnr;
   cout << "Ge namn : ";
   cin.getline(namn, sizeof(namn));
}
```

7)
```
void PERSON::skriv()
{
   cout << "Personens personnummer : ” << persnr << endl;
   cout << "Personens namn : ” << namn << endl;
}
```

8)
```
void main()
{
   PERSON a;

   a.las():
   if (a.is_girl())
      cout << ” Girl” << endl;
   else
      cout << ” Boy” << endl;
}
```

9)
```
bool PERSON::same_sex(PERSON p)
{
   return is_girl() && p.is_girl() || !is_girl() && !p.is_girl();
}
```

10)
```
int PERSON::get_alder()
{
   int persnrsum = 0, datesum = 0, sum;
   char date[9];

   for (int i = 0; i < 6; i++)
   {
```

```
        persnrsum = persnrsum * 10 + persnr[i] - '0';
    }

    get_date(date);

    for(int i = 2; i < 8; i++)
    {
      datesum = datesum * 10 + date[i] - '0';
    }

    if (persnrsum < datesum && persnr[6] == '-')
        sum = (datesum - persnrsum) / 10000;
    else if (persnrsum < datesum)
        sum = (datesum - persnrsum) / 10000 + 100;
    else
        sum = (datesum - persnrsum + 1000000) /10000;

    return sum;
}
```

11)

```
#include "Motor.h"

MOTOR::MOTOR (float varvtal)
{
   this->varvtal = varvtal;
}

float MOTOR::get_varvtal()
{
   return varvtal;
}


#include "Ventil.h"

VENTIL::VENTIL(float antal_varv)
{
   this->antal_varv = antal_varv;
}

float VENTIL::get_antal_varv()
{
   return antal_varv;
}


#include "Motorventil.h"

MOTORVENTIL::MOTORVENTIL(float varvtal,float antal_varv)
                         :m(varvtal), v(antal_varv)
{
}

float MOTORVENTIL::get_tid()
{
   return v.get_antal_varv() / m.get_varvtal();
}

#include "Motorventil.h"
#include <iostream.h>
#include <conio.h>

void main()
```

```
{
   MOTORVENTIL mv(20, 50);

   cout << "Tiden för att öppna ventilen : " << mv.get_tid() << endl;
   getch();
}
```

12)
```
PERSONAL::PERSONAL(char *persnr, char *namn, float lon)

                  : PERSON(persnr, namn)
{
   this->lon = lon;
}

void PERSONAL::las()
{
   PERSON :: las();
   cout << ”Ge lön : ”;
   cin >> lon;
}

void PERSONAL::skriv()
{
   PERSON::skriv();
   cout << ”Lön : ” << lon;
}

float PERSONAL::get_lon()
{
   return lon;
}

void main()
{
   PERSONAL a, b;

   a.las();
   b.las();

   if (a.get_lon() < b.get_lon())
   {
      a.skriv();
      b.skriv();
   }
   else
   {
      b.skriv();
      a.skriv();
   )
   getch();
}
```

13)
```
#include "Personal.h"
#include <conio.h>
#include <fstream.h>
#include <iostream.h>

void main()
{
   ifstream tsin("Personal.txt");
   int tjej_lon_sum = 0, kill_lon_sum = 0, lon;
   int nr_tjejer = 0, nr_killar = 0;
```

```
   char persnr[12], namn[30];


   while (tsin >> persnr >> namn >> lon)
   {
      PERSONAL  p(persnr, namn, lon);
      if (p.is_girl())
      {
         nr_tjejer++;
         tjej_lon_sum += p.get_lon();
      }
      else
      {
         nr_killar++;
         kill_lon_sum += p.get_lon();
      }
   }
   cout << "Tjejernas medellön = " << (float)tjej_lon_sum /
                                      nr_tjejer << endl;
   cout << "Killarnas medellön = " << (float)kill_lon_sum /
                                      nr_killar << endl;
   getch();
}
```

14)

```
#include "Biosalong.h"
#include <conio.h>
#include <string.h>
#include <iostream.h>


BIOSALONG:: BIOSALONG (int antal, char *namn)
{
   stolar = new bool[antal];
   for (int i = 0; i < antal; i++)
   {
      stolar[i]= false;
   }
   strcpy(this->namn, namn);
   this->antal = antal;
}

BIOSALONG::~BIOSALONG ()
{
   delete [] stolar;
}

bool BIOSALONG::boka(int radnr, int platsnr)
{
   int i = radnr * 10 + platsnr - 11;

   if (!stolar[i])
   {
       stolar[i] = true;
       return true;
   }
    return false;
 }

 void BIOSALONG::visa()
 {
    cout << namn << endl;
    for (int i = 0; i < antal; i++)
    {
```

```
        if (stolar[i])
           cout << " X " ;
        else
           cout << " 0 ";
        if ((i+1) % 10 == 0)
           cout << endl;
     }
  }
  int BIOSALONG::get_antal()
  {
     return antal;
  }
```

15)

```
class BIOSALONG
{
   private :
      . . . .

   public :
      . . . .
      BIOSALONG(const BIOSALONG &bio);
      const BIOSALONG &operator=(const BIOSALONG &bio);
      . . . .
};


BIOSALONG::BIOSALONG(const BIOSALONG &bio)
{
   stolar = new bool[bio.antal];
   for (int i = 0; i < bio.antal; i++)
   {
        stolar[i] = bio.stolar[i];
   }
   antal = bio.antal;
   strcpy(namn, bio.namn);
}


const BIOSALONG &BIOSALONG::operator=(const BIOSALONG &bio)
{
   if (this != &bio)
   {
      delete [] stolar;
      stolar = new bool[bio.antal];
      for (int i = 0; i < bio.antal; i++)
      {
         stolar[i] = bio.stolar[i];
      }
      strcpy(namn, bio.namn);
      antal = bio.antal;
   }
   return *this;
}
```